PeakML/mzMatch

Scheltema, Richard A.; Jankevics, Andris; Jansen, Ritsert C.; Swertz, Morris A.; Breitling, Rainer

Published in:
Analytical Chemistry

DOI:
10.1021/ac2000994

Publication date:
2011

analytical chemistry

ARTICLE

pubs.acs.org/ac

PeakML/mzMatch: A File Format, Java Library, R Library, and Tool-Chain for Mass Spectrometry Data Analysis

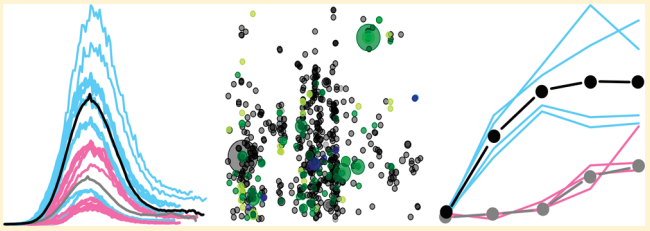
Richard A. Scheltema,[†] Andris Jankevics,[†,‡] Ritsert C. Jansen,[†] Morris A. Swertz,[*,†,§] and Rainer Breitling[*,†,‡]

[†]Groningen Bioinformatics Centre, Groningen Biomolecular Sciences and Biotechnology Institute, University of Groningen, Nijenborgh 7, 9747 AG Groningen, The Netherlands

[‡]Institute of Molecular, Cell and Systems Biology, College of Medical, Veterinary and Life Sciences, University of Glasgow, Joseph Black Building B3.10, G11 8QQ Glasgow, United Kingdom

[§]Genomics Coordination Center, Department of Genetics, University Medical Center Groningen and University of Groningen, P.O. Box 30001, 9700 RB Groningen, The Netherlands

ABSTRACT: The recent proliferation of high-resolution mass spectrometers has generated a wealth of new data analysis methods. However, flexible integration of these methods into configurations best suited to the research question is hampered by heterogeneous file formats and monolithic software development. The mzXML, mzData, and mzML file formats have enabled uniform access to unprocessed raw data. In this paper we present our efforts to produce an equally simple and powerful format, PeakML, to uniformly exchange processed intermediary and result data. To demonstrate the versatility of PeakML, we have developed an open source Java toolkit for processing, filtering, and annotating mass spectra in a customizable pipeline (mzMatch), as well as a user-friendly data visualization environment (PeakML Viewer). The PeakML format in particular enables the flexible exchange of processed data between software created by different groups or companies, as we illustrate by providing a PeakML-based integration of the widely used XCMS package with mzMatch data processing tools. As an added advantage, downstream analysis can benefit from direct access to the full mass trace information underlying summarized mass spectrometry results, providing the user with the means to rapidly verify results. The PeakML/mzMatch software is freely available at http://mzmatch.sourceforge.net, with documentation, tutorials, and a community forum.



Recent years have seen new and exciting metabolomics and proteomics experiments enabled by an increasing variety and improved performance of mass spectrometry equipment.[1,2] Standardization initiatives such as the mzXML file format[3] and the recently introduced mzML[4] have considerably shortened the development cycle for analysis software, and, as a consequence, a wealth of data analysis applications has become available, of which XCMS[5] and mzMine[6] are commonly used for metabolomics data sets. However, mzXML only standardizes the description of raw mass spectrometry data. Downstream integration of analysis tools into suitable analysis pipelines is still hindered by (i) use of monolithic, black box approaches where algorithms and resulting (intermediate) data are inaccessible to the user for verification, (ii) limited ability to check intermediate results of data processing and limited access to the underlying context information (e.g., peak shape or neighboring peaks), and (iii) data format heterogeneity at various steps of the analytic pipeline, making it difficult to recombine pipeline components to suit new experiments or technologies, e.g., when changes in chromatographic conditions or mass accuracy require different peak picking or filtering modules.

In the literature, a number of initiatives to standardize the storage of extracted features have already been described, including the initiatives developing FeatureXML[7] and CMLspect.[8] Even though these provide basic support for storage of extracted chromatogram information, they lack many of the options that PeakML offers. Additionally, some of their features, such as FeatureXML's comprehensive framework for storing protein/peptide identifications, create unnecessary overhead and restrictions in the context of metabolomics experiments.

In order to pick up where the current open formats leave off, we have developed the PeakML file format, an open and extensible format for the standardized representation of peak and meta-information from each step in a downstream analysis pipeline. The power of PeakML and mzMatch for rapid tool integration is demonstrated by a collection of small tools[9,10] and the availability of PeakML *read* and *write* functionality for XCMS,[5] a widely used data analysis software.[11−13] Equivalent converters can easily be created for other generic mass spectrometry processing tools. This comprehensive collection of components is intended to further encourage a modular and interchangeable design of analysis components, storing data generated/extracted by each step in a standardized manner. The added value for algorithm developers is that they can build

Received: January 13, 2011
Accepted: February 25, 2011
Published: March 14, 2011

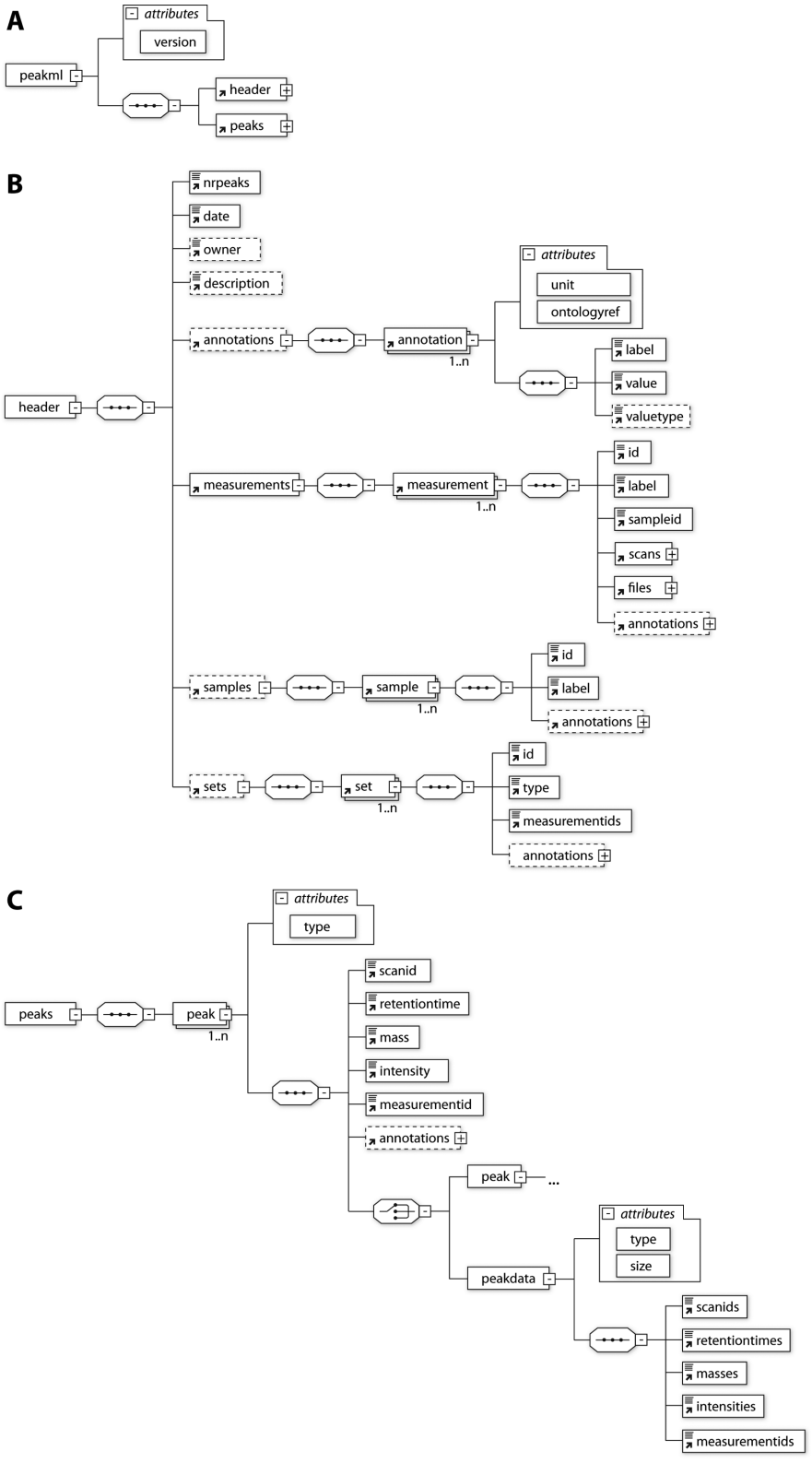


Figure 1. PeakML file format. (A) The format consists of two separate blocks: the header and the peaks block. (B) The header block stores general information about the contents of the file (e.g., date of creation). Beside this general information, full descriptions on the measurements are provided (e.g., such as ionization mode, original file, etc.) and how they are organized in sets. Each measurement also contains a measurement-id, which can be used to link the measurement data to the information stored in the peaks block. The contents of each section can be extended with annotations (label-value pairs) (C) The peaks block holds all intermediate and result mass spectrometry data. A peak is described by its mass, intensity, measurement-id, optional scan (for LC-MS data), and optional annotations for a peak. Each peak can be typed as being either *backgroundion* or *masschromatogram* (using the peakdata element), or as type *peakset* (using the recursive peak element).

dx.doi.org/10.1021/ac2000994 |*Anal. Chem.* 2011, 83, 2786–2793

Table 1. File Size Comparison

sample[a]			PeakML data (Mb)			number of features or peak sets in PeakML file		
cond	rep	mzXML data (Mb)	1	2	3	1	2	3
1	A	10.4	2.0	9.5	25.8	9504	16909	23926
1	B	12.3	2.3	9.5	25.8	10696	16909	23926
1	C	12.0	2.5	9.5	25.8	11867	16909	23926
2	A	10.1	2.1	9.7	25.8	10027	17556	23926
2	B	12.0	2.3	9.7	25.8	10897	17556	23926
2	C	12.2	2.5	9.7	25.8	12014	17556	23926
3	A	10.9	2.3	8.7	25.8	10865	15131	23926
3	B	12.1	2.1	8.7	25.8	9996	15131	23926
3	C	10.1	1.8	8.7	25.8	8742	15131	23926

[a] Samples were acquired in centroid mode on a Thermo LTQ-Orbitrap XL instrument. Binary native data were converted to the mzXML file format with ReAdW (a tool of the Trans-Proteomic Pipeline software collection, downloaded from http://tools.proteomecenter.org/wiki/index.php?title=Software:ReAdW). A final reduction of file size by 75% is achieved. cond = analytical condition; rep = biological replicate, 1 = PeakML file of the single LC/MS run after peak detection, 2 = PeakML file of the combined peak sets of biological replicates, 3 = PeakML file of combined peak sets between biological replicates and conditions.

on off-the-shelf PeakML software components (e.g., for data loading and visualization) and gain access to a potentially much larger user community for their tools.

APPLICATION PROGRAMMING INTERFACE

PeakML File Format. The PeakML file format was specifically designed to provide an open XML standard for the storage of hyphenated mass spectrometry data, of which a summary is given here. It differs from existing efforts such as mzML in that it supports common downstream results, ranging from storing mass chromatograms, background ions,[9] and any combination of either. In order to support the development of automatic processing software, a comprehensive metadata structure is provided. Information about the measurements (such as experimental parameters and machine settings) and how they are organized can be stored in this structure. The XML schema of the PeakML file format can be divided into two blocks (Figure 1A): a *header* block for storing the metadata and a *peaks* block for storing peak information.

The header block (Figure 1B) provides the structure for storing the required metadata for each step divided into four components (*measurements*, *samples*, *sets*, and *applications*), each of which can be extended with annotations supporting controlled vocabularies.[14−16] (i) The obligatory measurements block contains basic information about the measurements collected in the current analysis, such as names of all the files used, and a link to the sample description. (ii) The samples block contains information on the sample, which is kept to a basic level with an id, optional label, and annotations. (iii) Multiple measurements can be organized in a set or set of sets, e.g., all technical replicates for a certain sample are part of a single set, and the technical replicates for several related biological replicates are organized as a set of sets. The sets block contains information on how the measurements are organized in these sets, which can be used by automatic processing software, such as an RSD filter[17] that filters non-reproducible peaks across technical and/or biological replicates, and by visualization software, that assigns a single color to all peaks in a set of technical or biological replicates. (iv) The applications block contains information about the software components or 'steps' used to produce the file, e.g., a peak extraction tool would provide information such as its version, the raw data file, and the expected mass accuracy of the machine in parts-per-million. This allows for a complete reconstruction of the entire analysis protocol used and provides an archival trace of all raw and intermediate data files used to generate the current data set.

The peaks block (Figure 1C) provides the complete structure for storing information on one or multiple peaks. As the file format is focused on hyphenated mass spectrometry data, a peak is defined here as either a mass chromatogram, a background ion, or a set of one of these. Background ions[9] in PeakML are defined as analytes present over the whole retention time range (and are generally of no interest for the biological interpretation), while mass chromatograms (EICs) are caused by analytes eluting over a narrow retention time window (and are of interest for the biological interpretation).

The type of each peak entry is identified by the attribute type (which supports: masschromatogram, backgroundion, and peakset), providing an extensible construct for future versions and backward compatibility. Each entry is opened with summary information: 'scanid', 'retentiontime', 'mass', 'intensity', and 'measurementid', which can be used to load the entries as a flat peak table, without the complete trace information. More annotations can be stored for each peak entry, analogous to the header entries. The real data for each peak entry is made up of either a new peak entry (only for type 'peakset') or a *peakdata* entry containing the trace information (for 'masschromatogram' and 'backgroundion'). The trace information is stored as Base-64-encoded arrays in little-endian ordering (analogous to the mzXML and mzML formats), reducing the memory requirements. Currently only centroid single mass analyzer mode data are supported, but facilities for profile data are in place for future support. Support for additional mass spectrometry data, including tandem spectra, lies outside the aim of this format, as this is covered by other data formats.[18]

In order to minimize nongeneralizable, application-specific overhead (e.g., protein identification support) in the format, we decided to focus solely on extracted feature support. The concept of annotations provides an extensible platform to add additional identification information, for which two approaches can be taken: (1) Storage of the most pertinent information itself, e.g.,

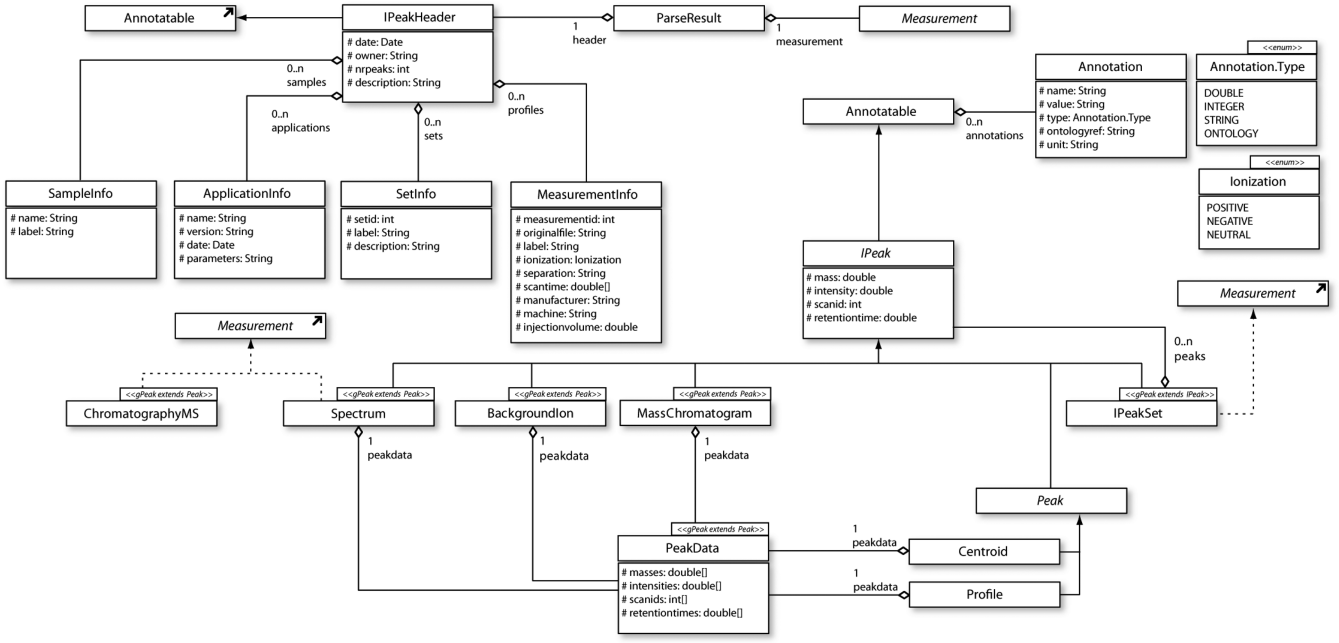


Figure 2. UML diagram of the mass spectrometry specific interface. The diagram contains two subsets, the mass spectrometry classes and the I/O classes. (1) The base-class is IPeak, which defines the basic properties of a signal or peak (mass, intensity, scan-id, and retention time, plus a measurement-id to link it to meta-information on the measurement). These properties are further specialized and extended in the subclasses (e.g., retention time for a mass chromatogram is defined as the time where the signal is most intense). The subclass *IPeakSet* copies and extends on the behavior of *java.util.Vector* by providing additional functionality on the list of peaks (e.g., fast binary search on IPeak properties). (2) Each parse method returns an instance of *ParseResult*, containing pointers to a Header instance and an instance of type Measurement (either Spectrum, ChromatographyMS, or IPeakSet), which was introduced to identify those classes that can be stored in a file. The class *Annotatable* (arguably the real base-class) injects functionality into classes for adding annotations (label−value pairs); this is disabled for the class *Peak* and all its derivatives to keep memory usage at a minimum.

as a descriptive text, or (2) storage of a link to an entry in a companion file containing the complete information (e.g., protein identification stored in an mzIdentML[18] file).

PeakML allows the effective storage of mass chromatograms (extracted ion chromatograms) for single or multiple measurements, achieving an average data reduction of around 75%. Typical file sizes for input data in mzXML format and PeakML data files are shown in Table 1. Examples of PeakML files are available for download at http://mzmatch.sourceforge.net/peakml_files.html.

We envision that PeakML will be used in combination with other formats that describe the whole experimental process (protocols, biomaterials, experimental variables, hypotheses, and conclusions). These are the domain of complementary data models such as FuGE,[19] MIAPE,[20] and XGAP.[21]

The PeakML Library. The PeakML Java library defines the fully documented Application Programming Interface (API) for handling PeakML data, parsers to load mass spectrometry files, and other components useful for building mass spectrometry analysis tools, including (but not limited to) chemistry, math, and user interface routines, considerably speeding up application development.

The core of the library consists of classes specific to easily interact with mass spectrometry data stored in raw and PeakML file formats (see Figure 2). The base class *IPeak* defines the minimal properties of each data element, such as mass, intensity, scan-number, and retention time; additionally a measurement-id can be defined to link an instance of IPeak to a *Measurement*. From IPeak commonly encountered mass spectrometry data types are extended including *Spectrum*, *MassChromatogram*, *BackgroundIon*, *ChromatographyMS*, and *PeakSet*. The class Spectrum defines a single scan of a mass spectrometer, either in continuous or centroid mode. The classes MassChromatogram and BackgroundIon represent a mass trace of an analyte in a hyphenated setup (i.e., GC-MS or LC-MS), extracted from a set of consecutive spectra. All MassChromogram objects from one experiment can be stored as a bundle in the class ChromatographyMS. Mass spectrometry data sets are generally quite large, which can cause problems with the inefficient memory usage of Java programming objects. To circumvent these limitations, the class *PeakData* was introduced, which has memory-efficient arrays for: scan numbers, retention times, masses, intensities, and measurement-ids. The classes Spectrum, MassChromatogram, and BackgroundIon use this class to store all the data, minimizing the memory requirements. To ease interaction with PeakData from the end-programmer, there are two helper classes *Centroid*, and *Profile* to easily interact with stored data.

Next to the data classes, a series of meta data 'header' classes are provided to describe the data sets. In addition a comprehensive set of I/O routines is provided for data loading and writing of mass spectrometry data in the major open file formats mzData, mzXML, mzML, next to PeakML. As each of these formats follows roughly the same format in terms of meta-information, they can be loaded into the common header classes. The class *Header* is the entry point, binding the following classes: *MeasurementInfo* with measurement-specific information, such as associated files, uniquely identifiable by a measurement-id also stored in IPeak; *SetInfo* that combines multiple measurements into a set;

dx.doi.org/10.1021/ac2000994 |*Anal. Chem.* 2011, 83, 2786–2793

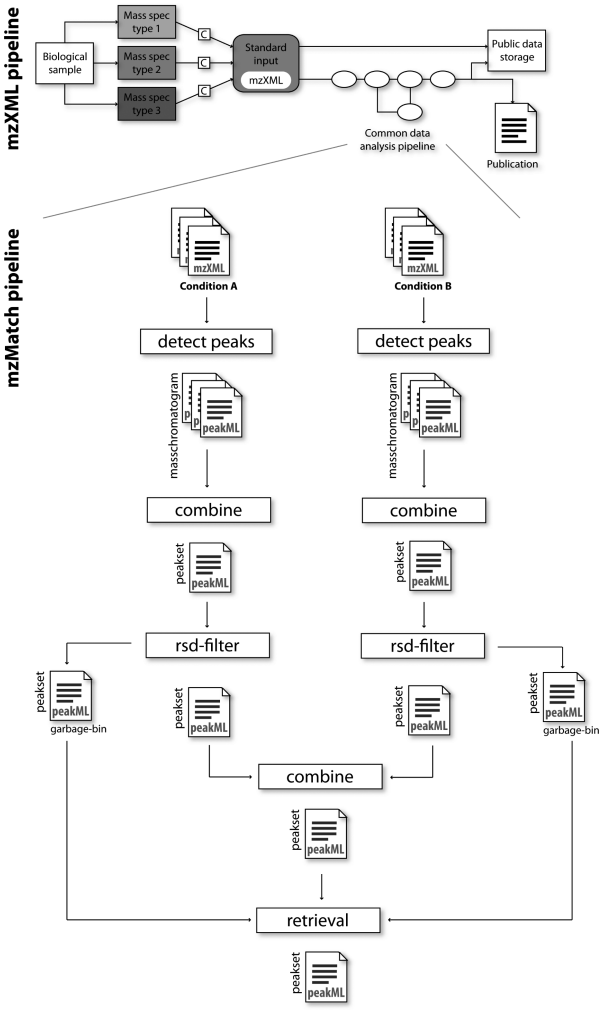


Figure 3. An example of a mzMatch pipeline. The mzXML/mzData/mzML standards changed the data analysis landscape by providing a common input format but do not provide functionality for breaking up a data analysis pipeline into small interchangeable components. The mzXML paper[3] proposed the pipeline shown at the top but did not go into further detail about the setup of a common data analysis pipeline. The mzMatch pipeline makes effective use of the PeakML file format for defining small components in a data analysis pipeline. The data in each file can be picked up by any tool; e.g., in this small example a noise filter could have been introduced right after the 'detect peaks' tool, without breaking the pipeline.

SampleInfo describes sample specific information, stored in annotations; *ApplicationInfo* describes software application-specific information that was used to produce the data, such as software name, version, parameters, etc. Additionally, 1-to-1 mappings are provided to the file access libraries of Waters Corp. and Thermo Fisher Scientific Inc., both of which are only accessible on Microsoft Windows platforms (due to the implementation chosen by these manufacturers).

The raw output file from a mass spectrometry run contains large amounts of meta- and status-information, which is rarely (if at all) preserved during the transformation process to an open standard such as mzXML. For further interpretation of the data, this meta-information can play a key role, providing insight into the quality of each measurement. For this reason, we additionally supply the tool ThermoLogViewer, which allows the user to load multiple RAW-files and compare their logs (see online documentation at http://mzmatch.sourceforge.net for more information).

The PeakML library allows for quick access to the content of data files. As a reference, we used the data files listed in Table 1. On a Dell Optiplex 780 desktop computer with an Intel Core2 Duo Q9550 CPU and 8 GB of RAM, loading PeakML files containing features for single LC/MS runs required 2 s. The initial loading and displaying of a massive PeakML file containing 23 926 peak sets required 13 s. Once loaded into memory, all extracted ion chromatograms can be accessed instantly.

The mzmatch.R R Package. The PeakML file format enables the uniform exchange of intermediate and result data between analysis software from different manufacturers and groups. This is important because each piece of software has its own unique strengths and weaknesses. Cherry-picking of components enables researchers to construct a data analysis pipeline specifically suited to their needs. Use of PeakML can enable such flexible pipelines, which we illustrate here by integrating PeakML with the R-package XCMS.[5] This package has excellent support for data processing, statistics, and graph visualization. However, R is primarily targeted at programmers, potentially locking nonexpert users out from further data analysis. Moreover, it is difficult to visualize the extracted peaks such that one can browse through them and select the peaks of interest (e.g., the getEIC routines of XCMS are not straightforward to apply). In contrast, this functionality is easily implemented in the framework with the PeakML file format, as shown with the 'PeakML Viewer'. The R-package 'mzmatch.R' extends XCMS with functionality for storing data in PeakML files and vice versa, such that these tools can be connected (see online documentation at http://mzmatch.sourceforge.net for more information). Because PeakML includes system-defined annotations (e.g., 'identifications': a comma-separated list of database id's; 'relation.id' and 'relation.ship': identifiers for derivative peaks and their relationships), visualizations beyond the current capabilities of XCMS are enabled.

APPLICATIONS

mzMatch. The availability of the PeakML file format makes it possible to split the components of a processing pipeline into small tools (peak extraction, alignment, noise filtering, etc., described in refs 9 and 10) that can easily be connected into various configurations. This principle has been applied to the implementation of the data analysis pipeline mzMatch (Figure 3; an extensive step-by-step tutorial on the example pipeline is available at http://mzmatch.sourceforge.net/tutorial.mzmatch.r.advanced.html). The tools included are mass chromatogram extraction, matching (called "grouping" in XCMS), derivative detection,[10] noise filtering, normalization, and alignment (see online documentation at http://mzmatch.sourceforge.net/mzmatch/index.html and http://mzmatch.sourceforge.net/mzmatch.R/00Index.html). For example, the "RelatedPeaks"[10] tool is very effective in gathering all features caused by a single analyte (including isomers) and annotating them accordingly. This means the features are not removed from the data, but only tagged, allowing for later inspection by the analyst. It is the responsibility of the analyst to validate the identity of the peaks with additional, orthogonal biochemical techniques or internal standards.

In addition, all filter tools discarding signals from the data set, such as the CoDA-DW[22] noise filter *mzmatch.filter.NoiseFilter*,

also export the discarded signals next to the result data set. Such behavior provides traceability for the performance of each tool, as the user can easily verify whether the operation had the desired effect. It also offers the potential for retrieving signals of interest lost in one of the filter steps which cannot easily be achieved with other data processing packages. The RSD filter,[17] *mzmatch.filter.RSDFilter*, removes signals that were irreproducible in biological and/or technical replicates. However, when multiple experimental conditions are used, it can happen that the behavior for signals in one condition falls outside the specified range of reproducibility, but not in the other(s). Such signals can then be recovered in all experiments with the recycle bin recovery tool, *mzmatch.util.Recovery*.

As described previously, the output of each tool can readily be used in either XCMS or the mzMatch/PeakML Viewer. Each tool is command-line-based such that settings can be passed to the tools as command-line options (e.g., '-i' for the input file(s)). The mzMatch toolbox is designed in such a way that documentation for each tool can automatically be generated. In addition, tools can also be automatically exposed in a programming language (such as the R environment) as functions or a pipeline workflow environment such as Taverna.[23] This has been done to extend the mzmatch.R library with the mzMatch tools.

PeakML Viewer. A user interface application called PeakML Viewer (Figure 4) enables rapid visualization, inspection, and manipulation of the contents of a PeakML file (e.g., manual selection and/or export of peaks of interest). After a PeakML file is loaded, the "entry" view gives an overview of all the entries with the retention time, mass, and intensity. The ordering from the original file is kept intact, making the results from sorting tools such as *mzmatch.ipeak.sort.RelatedPeaks*[10] accessible. An entry is highlighted in bold when it has been matched to a compound from a database with *mzmatch.ipeak.util.Identify* (determined by verifying whether the entry contains the system-defined annotation *identification*). By clicking on an entry, the associated traces will be displayed in the graph view and the identifications in the "identification" view (including the mass deviation in ppm and the putatively assigned molecular structure when available). There is an additional tab *derivatives*, which shows all the clustered related peaks with their identification as determined by *mzmatch.ipeak.sort.RelatedPeaks* (stored in the system defined annotations *relation.id* and *relation.ship*). The "filter" view allows the user to perform simple operations on the data (sorting and filtering), for zooming in on the entries of interest. The "trend" view gives an overview of the intensity levels for each entry (which in the case of a peakset can for example consist of multiple mass chromatograms).

All the peaks belonging to the same set are grouped together in this plot, and the mean, minimum, and maximum values are displayed. The "sets" view shows all the measurements used and how they are organized in sets. With the check boxes, all the peaks from a set or peaks individually can be switched on or off (i.e., not displayed). The "annotations" view gives an overview of all the annotations that are available for the current entry.

DISCUSSION

The PeakML file format enables research groups to transcend the monolithic development model of mass spectrometry data analysis software and start building flexible, modular application

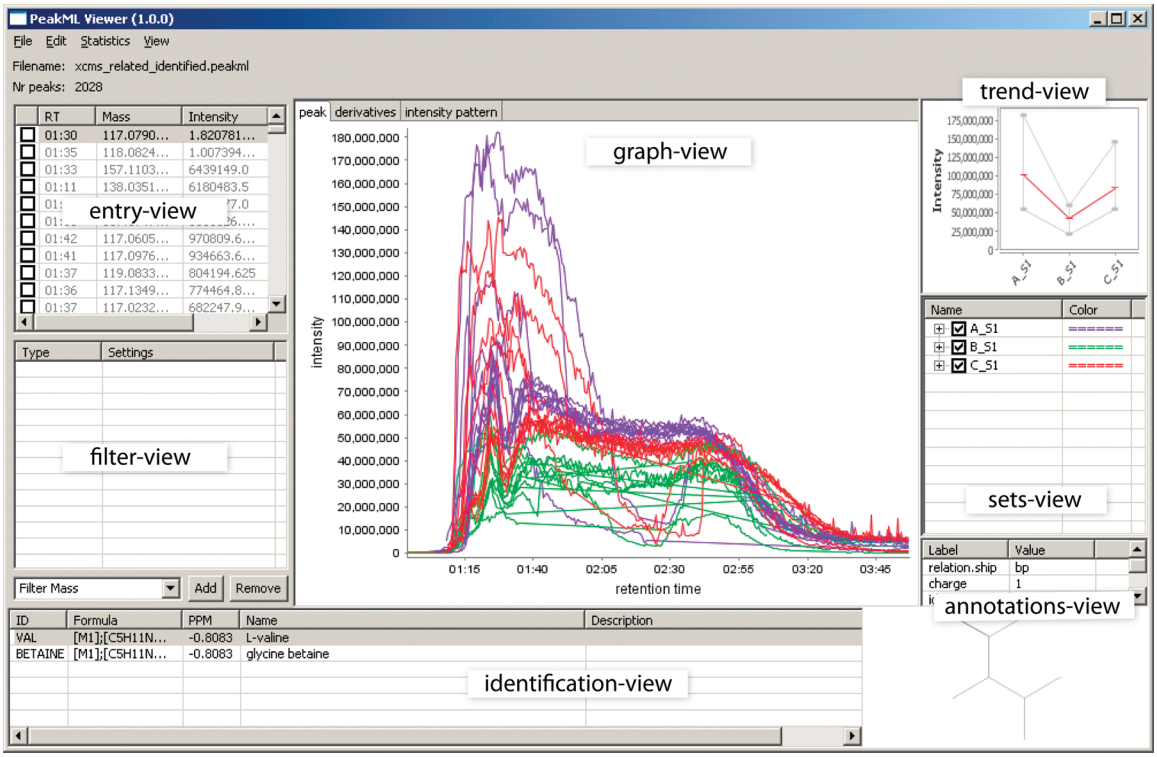


Figure 4. Screenshot of the PeakML Viewer. The viewer enables the user to load PeakML files (generated at any point in the analysis pipeline, providing full control and verifiability of the data processing steps), visualize, and browse through its contents. The interface is divided into seven views, providing all the information stored in the file. The most important are the "entry" view, giving an overview of all the IPeak entries stored in the file, and the "graph" view, giving a visual representation of the data stored for the entry. By using the keyboard (arrow up/down and spacebar for selection), the user can rapidly verify the contents and select peaks of interest.

dx.doi.org/10.1021/ac2000994 |*Anal. Chem.* 2011, 83, 2786–2793

pipelines. The benefits include (i) increased verifiability of the performance of individual analysis steps, (ii) an easy "rewind" option to roll back to intermediate steps in the analysis process, and (iii) the straightforward use of analytical components from alternative pipelines originally not intended by the software authors. Moreover, tool developers can have a much broader user group for their software, because its components can be more easily recombined to suit the needs of different researchers. With the PeakML and mzmatch.R libraries, a first successful integration between data analysis environments created by different groups has been demonstrated. Of course, PeakML still has limitations. For example, both PeakML and the mzMatch toolbox have been developed mostly for metabolomics experiments,[24−26] but their functionality for proteomics experiments is in its infancy. Additionally, adding an indexing mechanism to speed up searches in large PeakML files is desirable. We would like to invite other groups to join in this open source development at http://mzmatch.sourceforge.net to enable the larger mass spectrometry community to share the best data and tools notwithstanding large variation in research aims.

PeakML/mzMatch highlights:

- Fully documented, complete, and platform-independent mass spectrometry specific API complete with programming examples.
- Defines the PeakML file format, offering functionality to store and share processed data for further processing and to publish verifiable results.
- Supports the major file formats mzData, mzXML, mzML; provides a 1-to-1 mapping to the file access libraries of Waters Corp. and Thermo Fisher Scientific Inc.
- Plays well with others by offering the potential to integrate with other software, as illustrated by the integration with XCMS.
- Integrated chemistry (e.g., molecular formulas, mass conversion, periodic table), math (e.g., statistics, wavelet transform, function fitting, and loess and Savitzky−Golay), and visualization (JFreeChart and SWT for user interface applications) routines.
- A set of small and agile tools (e.g., mass chromatogram extraction, combining, noise filtering, normalization) performing defined operations on the data.

AUTHOR INFORMATION

Corresponding Author

*E-mail: (R.B.) rainer.breitling@glasgow.ac.uk; (M.A.S.) m.a.swertz@rug.nl.

Notes

Competing Interests

The authors have declared that no competing interests exist.

ACKNOWLEDGMENT

R.A.S. and A.J. contributed equally to this work. The authors gratefully acknowledge the efforts and contributions of Elena Merlo and George Byelas (University of Groningen, The Netherlands), Saskia Decuypere and Ruben t'Kindt (Institute of Tropical Medicine, Belgium), Martijn Dijkstra and Marcel de Vries (University Medical Centre Groningen, The Netherlands), David Wildridge, Jana Anderson, Isabel May-Vincent, Darren Creek, Karl Burgess, and Michael Barrett (University of Glasgow, UK), Fabien Jourdan (INRA, France), and Anas Kamleh and David Watson (Strathclyde University, UK) for providing data, testing the software, and suggesting improvements and additional features. R.B. and R.C.J. devised and supervised the project. R.A.S. designed and implemented the software architecture and the PeakML file format. A.J. designed and implemented the XCMS integration and R-tools accompanying the software. R.S., A.J., M.A.S., and R.B. wrote the manuscript. R.B. is supported by a Netherlands Organisation for Scientific Research (NWO) Vidi fellowship. R.A.S. is supported by a NWO Vici grant to RCJ. M.A.S. is supported by NWO Rubicon (825.09.008) and The Netherlands Bioinformatics Center. R.B. and M.A.S. are supported by The Netherlands Proteomics Center (NPC-GM WP 1.1).

REFERENCES

(1) Dunn, W. B. *Phys. Biol.* **2008**, *5*, 11001.

(2) Han, J.; Danell, R. M.; Patel, J. R.; Gumerov, D. R.; Scarlett, C. O.; Speir, J. P.; Parker, C. E.; Rusyn, I.; Zeisel, S.; Borchers, C. H. *Metabolomics* **2008**, *4*, 128–140.

(3) Pedrioli, P. G. A.; Eng, J. K.; Hubley, R.; Vogelzang, M.; Deutsch, E. W.; Raught, B.; Pratt, B.; Nilsson, E.; Angeletti, R. H.; Apweiler, R.; Cheung, K.; Costello, C. E.; Hermjakob, H.; Huang, S.; Julian, R. K.; Kapp, E.; McComb, M. E.; Oliver, S. G.; Omenn, G.; Paton, N. W.; Simpson, R.; Smith, R.; Taylor, C. F.; Zhu, W.; Aebersold, R. *Nat. Biotechnol.* **2004**, *22*, 1459–1466.

(4) Deutsch, E. W. In *Proteome Bioinformatics*; Humana Press: Totowa, NJ, 2010; pp 319−331.

(5) Smith, C. A.; Want, E. J.; O Maille, G.; Abagyan, R.; Siuzdak, G. *Anal. Chem.* **2006**, *78*, 779–787.

(6) Pluskal, T.; Castillo, S.; Villar-Briones, A.; Oresic, M. *BMC Bioinf.* **2010**, *11*, 395.

(7) Sturm, M.; Kohlbacher, O. *J. Proteome Res.* **2009**, *8*, 3760–3763.

(8) Kuhn, S.; Helmus, T.; Lancashire, R. J.; Murray-Rust, P.; Rzepa, H. S.; Steinbeck, C.; Willighagen, E. L. *J. Chem. Inf. Model.* **2007**, *47*, 2015–2034.

(9) Scheltema, R. A.; Kamleh, A.; Wildridge, D.; Ebikeme, C.; Watson, D. G.; Barrett, M. P.; Jansen, R. C.; Breitling, R. *Proteomics* **2008**, *8*, 4647–4656.

(10) Scheltema, R.; Decuypere, S.; Dujardin, J.; Watson, D.; Jansen, R.; Breitling, R. *Bioanalysis* **2009**, *1*, 1551–1557.

(11) Arbona, V.; Iglesias, D. J.; Talón, M.; Gómez-Cadenas, A. *J. Agric. Food Chem.* **2009**, *57*, 7338–7347.

(12) Dai, Y.; Li, Z.; Xue, L.; Dou, C.; Zhou, Y.; Zhang, L.; Qin, X. *J. Ethnopharmacology* **2010**, *128*, 482–489.

(13) Lin, H.-M.; Edmunds, S. J.; Helsby, N. A.; Ferguson, L. R.; Rowan, D. D. *J. Proteome Res.* **2009**, *8*, 2045–2057.

(14) Bodenreider, O. *Nucleic Acids Res.* **2004**, *32*, D267–270.

(15) Stoeckert, C. J.; Parkinson, H. *Comp. Funct. Genom.* **2003**, *4*, 127–132.

(16) Whetzel, P. L.; Brinkman, R. R.; Causton, H. C.; Fan, L.; Field, D.; Fostel, J.; Fragoso, G.; Gray, T.; Heiskanen, M.; Hernandez-Boussard, T.; Morrison, N.; Parkinson, H.; Rocca-Serra, P.; Sansone, S.-A.; Schober, D.; Smith, B.; Stevens, R.; Stoeckert, C. J.; Taylor, C.; White, J.; Wood, A. *Omics* **2006**, *10*, 199–204.

(17) Shah, V. P.; Midha, K. K.; Findlay, J. W.; Hill, H. M.; Hulse, J. D.; McGilveray, I. J.; McKay, G.; Miller, K. J.; Patnaik, R. N.; Powell, M. L.; Tonelli, A.; Viswanathan, C. T.; Yacobi, A. *Pharm. Res.* **2000**, *17*, 1551–1557.

(18) Eisenacher, M. In *Data Mining in Proteomics*; Humana Press: Totowa, NJ, 2011; pp 161−177.

(19) Jones, A. R.; Lister, A. L. *Methods Mol. Biol.* **2010**, *604*, 333–343.

(20) Taylor, C. F.; Paton, N. W.; Lilley, K. S.; Binz, P.-A.; Julian, R. K., Jr.; Jones, A. R.; Zhu, W.; Apweiler, R.; Aebersold, R.; Deutsch, E. W.; Dunn, M. J.; Heck, A. J. R.; Leitner, A.; Macht, M.; Mann, M.; Martens, L.; Neubert, T. A.; Patterson, S. D.; Ping, P.; Seymour, S. L.;

Souda, P.; Tsugita, A.; Vandekerckhove, J.; Vondriska, T. M.; Whitelegge, J. P.; Wilkins, M. R.; Xenarios, I.; Yates, J. R., III; Hermjakob, H. *Nat. Biotechnol.* **2007**, *25*, 887–893.

(21) Swertz, M. A.; Velde, K. J.; van der; Tesson, B. M.; Scheltema, R. A.; Arends, D.; Vera, G.; Alberts, R.; Dijkstra, M.; Schofield, P.; Schughart, K.; Hancock, J. M.; Smedley, D.; Wolstencroft, K.; Goble, C.; de Brock, E. O.; Jones, A. R.; Parkinson, H. E.; Jansen, R. C. *Genome Biol.* **2010**, *11*, R27.

(22) Windig, W. *Chemom. Int. Lab. Syst.* **2004**, *77*, 206–214.

(23) Kuhn, T.; Willighagen, E. L.; Zielesny, A.; Steinbeck, C. *BMC Bioinf.* **2010**, *11*, 159.

(24) Kol, S.; Merlo, M. E.; Scheltema, R. A.; de Vries, M.; Vonk, R. J.; Kikkert, N. A.; Dijkhuizen, L.; Breitling, R.; Takano, E. *Appl. Environ. Microbiol.* **2010**, *76*, 2574–2581.

(25) t'Kindt, R.; Scheltema, R. A.; Jankevics, A.; Brunker, K.; Rijal, S.; Dujardin, J.-C.; Breitling, R.; Watson, D. G.; Coombs, G. H.; Decuypere, S. *PLoS Negl. Trop. Dis.* **2010**, *4*, e904.

(26) t'Kindt, R.; Jankevics, A.; Scheltema, R. A.; Zheng, L.; Watson, D. G.; Dujardin, J.-C.; Breitling, R.; Coombs, G. H.; Decuypere, S. *Anal. Bioanal. Chem.* **2010**, *398*, 2059–2069.